Optimizing AI Decision-Making in Tank Battles Using Deep Q-Learning

Hikaru Isayama
University of California San Diego
hisayama@ucsd.edu

Avi Mehta
University of California San Diego
avmehta@ucsd.edu

Wilson Liao
University of California San Diego
wil011@ucsd.edu

Julia Wong
University of California San Diego
juw024@ucsd.edu

Abstract

Deep Q-Learning (DQL) has demonstrated success in a variety of reinforcement learning tasks, particularly in game-playing and control settings. In this study, we explore the effectiveness of DQL in an adversarial tank battle game where an agent must learn to aim, shoot, and evade an opponent. We design two experiments: one where the agent learns to aim at a stationary target, and another where it must dynamically balance aiming and dodging against a moving adversary. Our results indicate that while the agent demonstrated some strategic behavior, its performance remained inconsistent, with high volatility in Q-value estimates and reward progression. Against a random opponent, the trained model won 6 out of 10 games, but it failed to win against a heuristic opponent in 10 matchups. Against human players, the heuristic bot won 8 out of 10 games, while the DQL model secured only 2 wins. These results highlight the limitations of DQL in dynamic adversarial settings and suggest that improvements such as reward shaping, curriculum learning, and alternative architectures (e.g., PPO or CNN-based encoders) may be necessary to enhance stability and generalization.

1 Introduction

Deep Reinforcement Learning (DRL) has revolutionized the field of artificial intelligence by enabling agents to learn complex decision-making tasks through interaction with their environment. From mastering Atari games, to achieving superhuman performance in board games like Go, and excelling in real-time strategy games, DRL has demonstrated its ability to handle high-dimensional state spaces and strategic decision-making.

The breakthroughs in these domains highlight the strengths of DRL in diverse applications. The research on Atari (1) introduced deep Q-networks (DQN) to learn control policies directly from raw pixel input, achieving human-level performance in multiple games without prior domain knowledge. This work showcased the ability of DRL to generalize across different environments and laid the foundation for reinforcement learning in visual and dynamic settings. Similarly, the research on AlphaGo (2) combined deep neural networks with Monte Carlo Tree Search, achieving superhuman performance in the game of Go. This work demonstrated the potential of reinforcement learning in high-dimensional strategic decision-making problems and marked a milestone in AI's ability to outperform human experts in complex board games. In the realm of real-time strategy, AlphaStar (3) successfully applied multi-agent reinforcement learning to the game StarCraft II, showcasing DRL's capacity for long-term planning, micro-level control, and adaptation in partially observable environments.

The successes achieved in these games through deep reinforcement learning showcase its potential for learning control policies directly from raw pixel input, laying the foundation for its application in more complex environments. However, the application of DRL to adversarial and combat-based scenarios remains a challenging and active area of research. In many real-world and simulated competitive environments, autonomous agents must balance multiple objectives, such as avoiding threats while attacking opponents. These tasks require strategic planning, precise execution, and dynamic adaptation to an unpredictable opponent's behavior. Developing AI that can efficiently learn and generalize in such adversarial settings is crucial for applications ranging from video games and robotics to military simulations and autonomous defense systems.

This study focuses on optimizing decision-making in a competitive tank battle environment using Deep Q-Learning (DQL), a widely used reinforcement learning technique for discrete action spaces. Our goal is to train an agent that can effectively navigate, aim, and shoot while avoiding enemy attacks. The primary motivation behind this research is to explore how DRL-based approaches can outperform traditional heuristic strategies and random policies in high-stakes adversarial settings. Furthermore, our experiments shed light on the learning dynamics of DRL in combat-based simulations and provide insights into effective training methodologies for such environments.

By investigating the feasibility and performance of DQL in a simplified tank battle game, we contribute to the broader understanding of how deep reinforcement learning can be leveraged for real-time decision-making in competitive scenarios. Our findings provide a foundation for future research in optimizing reinforcement learning strategies for more complex multi-agent combat simulations and adversarial planning tasks.

2 Related Work

Deep reinforcement learning has significantly advanced artificial intelligence by enabling agents to develop decision-making policies in high-dimensional state spaces. This section examines the key factors and methodologies that have contributed to DRL's success, including Deep Q-Learning (DQL), Dueling Network Architectures, Proximal Policy Optimization (PPO), Imitation Learning, and heuristic approaches. Each of these techniques plays a crucial role in reinforcement learning, particularly in adversarial environments where agents must adapt to dynamic and competitive settings, which align with the goals of our project.

One of the most well-established reinforcement learning methods is Deep Q-Learning (DQL), which builds upon traditional Q-learning by incorporating deep neural networks for function approximation. This enables agents to make decisions in complex environments with large state spaces. Beyond the application to Atari games (1) covered in the section prior, DQL has been refined with various improvements over the recent years. Techniques such as prioritized experience replay (4) improve sample efficiency by prioritizing valuable experiences for training, while double Q-learning (5) mitigates overestimation bias in action-value predictions. Despite these advancements, DQL still struggles with stability and efficiency, especially in competitive and multi-agent settings where agent interactions make learning more unpredictable.

To address some of DQL's limitations, dueling network architectures introduce a further enhancement by separately estimating state values and action advantages. Dueling network architectures (6) aim to improve learning efficiency, particularly in environments where certain actions have minimal influence on overall performance. By decomposing the Q-function into state value and advantage components, the dueling network architecture allows agents to prioritize learning about important states rather than focusing on precise action values. This leads to improved efficiency and more stable learning. In the context of our tank battle environment, the dueling network structure enables the agent to recognize and prioritize strategic positions, such as taking cover, aiming at an opponent, or avoiding incoming fire. Since not every action (e.g., minor adjustments in movement) directly impacts the outcome, focusing on valuable states helps the agent develop more effective long-term combat strategies.

Another reinforcement learning approach that we chose to experiment with is Proximal Policy Optimization (PPO), which emerged as an alternative to value-based methods like DQL, particularly in continuous control and multi-agent reinforcement learning scenarios. Unlike DQL, which learns a value function, PPO directly optimizes policies using policy gradients while maintaining stability through clipped objective functions (7). PPO has been widely adopted for learning in dynamic environments, including robotics and multi-agent interactions (11), where it often outperforms traditional Q-learning in complex decision-making tasks. In the context of a tank battle environment, PPO's policy-based approach could enable a more flexible and adaptive decision-making compared to value-based methods. Since adversarial combat requires real-time adaptation to an opponent's movements and actions, PPO's ability to optimize policies continuously could allow the agent to develop more sophisticated evasion, aiming, and attack strategies. Moreover, its stability in policy updates ensures the agent can generalize well across different battle scenarios, improving long-term performance against dynamic opponents.

Another technique that has been explored to improve learning efficiency is Imitation Learning, which accelerates reinforcement learning by initializing policies with expert demonstrations. Unlike reinforcement learning, which depends on reward signals for exploration, imitation learning enables agents to directly learn from expert actions, making training more efficient. Augmented Q Imitation Learning (AQIL) (8) extends DQL by incorporating imitation-based pretraining, enabling faster convergence in complex environments. Similarly, Soft Q Imitation Learning (SQIL) (9) reframes imitation learning as reinforcement learning with sparse rewards, bridging the gap between supervised imitation and autonomous learning. In our tank battle environment, imitation learning can be leveraged to train the agent using expert demonstrations, such as human-controlled gameplay or scripted strategies. By pretraining the agent with expert knowledge, imitation learning can help the agent learn effective positioning, movement, and attack patterns more efficiently, reducing the need for extensive trial-and-error exploration.

Finally, heuristic-based strategies play an important role in reinforcement learning by providing structured guidance to improve learning efficiency. Traditional heuristic methods rely on predefined rules, but recent research has combined heuristics with reinforcement learning to enhance sample efficiency (10). In adversarial scenarios, heuristic opponents are frequently used to benchmark reinforcement learning agents before transitioning to fully learned adversarial play. In a tank battle environment, heuristic strategies play a critical role in shaping the early learning process. By training against predefined heuristic opponents, the agent can establish baseline strategies such as movement patterns, defensive positioning, and attack timing. This structured progression allows the agent to gradually transition from learning against rule-based adversaries to adapting against dynamic, self-improving opponents, ultimately enhancing its capacity for real-time decision-making in adversarial combat.

3 Methods

3.1 Game Environment

The tank battle game is a grid-based, discrete-time environment with a square board. The agent (green tank) starts at the center of the board, and the hueristic tank (red) start at random positions. Both tanks have three health points and can move in four directions (up, down, left, right) while independently adjusting their gun barrel direction. The agent's goal is to reduce the opponent's health to zero before its own health is depleted. The opponent follows a random policy, moving and firing unpredictably.

The game has several key mechanics that shape how the tanks interact. Each tank has a turret that can rotate independently from its movement, allowing for flexible aiming and firing. When a tank shoots, the projectile travels in a straight line and deals damage if it hits the opponent. However, a cool down system prevents rapid-fire shooting, introducing a layer of strategy where precise timing is essential. Tanks are restricted within the game boundaries and cannot move beyond the playable area (Figure 1).

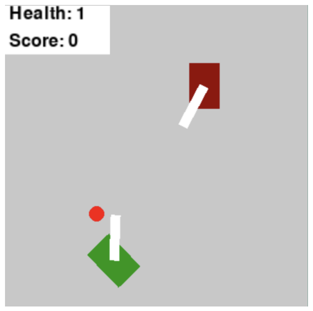

Figure 1: Instance of game environment made from scratch with pygame.

The environment progresses in discrete time steps, meaning actions such as moving, shooting, and turret rotation occur in fixed increments. The green tank, which is trained using reinforcement learning, must learn to aim accurately and position itself strategically while avoiding reckless movement. The opponent tank follows a random policy, making each encounter unpredictable and requiring the green tank to continuously adapt its strategy.

3.2 Heuristic Agent Strategy

In order to evaluate the Q Learning agent, we implemented a heuristic agent that operates with the following decision rules in the interval of an integer skill level. In practice, for a skill level of s, the agent acts according to the following rules once every s steps of uniformly-random sampled actions.

Barrel: The bot continuously tracks the player's position and rotates its gun barrel to align with the player. If the play is within ± 2 degrees of the heuristic agent's gun barrel, the agent fires.

Movement: After shooting, it selects a new random movement direction and moves accordingly to maintain elusive to the opponent.

3.3 Training Approach and Experimental Design

To evaluate the performance of our reinforcement learning agent, we designed two sequential experiments that incrementally increase in complexity. Both experiments are trained using Deep Q-Learning (DQL), a method that combines Q-learning with a deep neural network to estimate Q-values. These Q-values represent the expected future rewards for state-action pairs, enabling the agent to make strategic decisions. Instead of maintaining a Q-table, which is infeasible in high-dimensional spaces, the neural network generalizes learning across similar states, allowing for scalable and flexible decision-making.

State Representation: In both experiments, the environment is encoded into a fixed-size state vector consisting of:

- The positions of both tanks.
- The remaining health of each tank.
- The current facing direction of each tank.

This representation enables the agent to assess its surroundings and plan movements, aiming, and shooting strategies accordingly.

The agent is trained using DQL with experience replay. During training, transitions (s, a, r, s') are stored in a replay buffer. At each learning step, minibatches are sampled from the buffer to update the Q-network. The loss function is given by:

$$L = \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)^2,$$

where γ is the discount factor used to balance immediate versus future rewards.

For our exploration strategy, we decided to use an epsilon-greedy strategy to promote exploration. With probability ϵ, the agent selects a random action; otherwise, it selects the action with the highest estimated Q-value. The value of ϵ is gradually decayed throughout training, allowing the agent to shift from exploration to exploitation over time.

3.4 Experiment 1: Stationary Tanks Aiming Task

In our first experiment, both tanks remain stationary, and the agent (green tank) is trained solely on aiming. This experiment serves as a proof of concept. This experiment isolates the aiming component of the learning task. The green tank is fixed in position, and the opponent (red tank) is also stationary. The agent's objective is to align its turret and shoot accurately. The action space for this experiment is shown in Table 1

Table 1: Action Space for Experiment 1

Action	**Description**
Rotate Shooter Left	Adjusts turret counterclockwise
Rotate Shooter Right	Adjusts turret clockwise
Shoot	Fires a bullet (subject to cooldown)
Idle	No action taken

The network architecture used to estimate Q-values in this experiment is a simple feedforward neural network with one hidden layer of 256 units followed by ReLU activation, and a final linear output layer corresponding to each available action. This is summarized in Table 2.

Table 2: Model Architecture (Keras DQN)

Layer	**Size / Details**
Input (Flatten)	Shape: (1, `observation_dim`)
Dense Layer 1	256 units, ReLU activation
Output Layer	`n_actions`, Linear activation

Training parameters, such as learning rate, optimizer type, memory configuration, and exploration strategy, are detailed in Table 3. These values were chosen based on preliminary experiments and standard practices in DQL.

Table 3: Training Hyperparameters (Dueling DQN)

Parameter	**Value / Details**
Episodes	10,000,000
Test Episodes	5
Replay Memory	Limit = 1,000,000; window_length = 1
Warmup Steps	2,000
Target Model Update	0.01 (soft update)
Double DQN	Enabled
Exploration Policy	Linear Annealed Epsilon-Greedy
	ϵ: decays from 1.0 to 0.1 over first 10% of training steps
	$\epsilon_{\text{test}} = 0.05$
Policy	Epsilon-Greedy
Memory Type	SequentialMemory
Memory Limit	1,000,000 transitions
Optimizer	RMSprop
Learning Rate	0.001
Model Architecture	Dueling DQN with [256] hidden units
Loss Function	Mean Squared Error (MSE)

The reward structure in Experiment 1 encourages accurate aiming and quick resolution. Table 4 outlines the complete reward function used in this experiment.

Table 4: Reward Function Components (Experiment 1)

Condition	**Reward / Penalty**
Player Hits Enemy	$+5000$
Player Gets Hit by Enemy	-5000
Player's Barrel is x Degree Away From Enemy*	$-tanh(5 \times (0.5 - \frac{x}{180}))$
Player's Bullet is Shot x Degree Away From Enemy*	$-tanh(5 \times (0.5 - \frac{x}{180}))$
Winning Reward (Scaled by Time)	100((Player Health $\times 10$) $-1.0\times$ Steps)
Losing Penalty (Scaled by Time)	$-100(($Enemy Health $\times$ $10) - 1.0\times$ Steps)

*Note that this could also make a positive change to the reward if $x < 90$ to encourage aiming and discourage missing with a non-linear transition.

Formation of this reward function lies in the aiming mechanism. Since the model will be trained stationarily against another stationary enemy, the reward function poses to heavily encourage learning of effective aiming and shooting. Since this reward is given to the model cumulatively, the other rewards are scaled up to match the magnitude.

3.4.1 Experiment 2: Aiming and Dodging

In Experiment 2, we extend the agent's capabilities by introducing movement and a dynamic opponent. This version of the environment more closely resembles realistic combat, requiring the agent to aim, shoot, and dodge incoming fire simultaneously. This experiment uses a more sophisticated dueling DQN architecture and a richer action space to enable more complex behavior. However, while training we encountered computational issues running the dueling DQN architecture on our local machine, at which point we created and trained a standard DQN model that we were able to train for an extended period of time.

Table 5 outlines the expanded action space used in Experiment 2.

Table 5: Action Space for Experiment 2

Action	**Description**
Move Up	Moves tank upward
Move Down	Moves tank downward
Move Left	Moves tank left
Move Right	Moves tank right
Rotate Shooter Left	Adjusts turret counterclockwise
Rotate Shooter Right	Adjusts turret clockwise
Shoot	Fires a bullet (subject to cooldown)
Idle	No action taken

The dueling DQN model architecture is detailed in Table 7. This architecture separates the estimation of state-value and action-advantage functions, which helps stabilize learning and focus representation learning on important features.

Table 6: Model Architecture (Standard DQN)

Layer	**Size / Details**
Input (Flatten)	Shape: (1, `observation_dim`)
Dense Layer 1	24 units, ReLU activation
Dense Layer 2	24 units, ReLU activation
Output Layer	`n_actions`, Linear activation

Table 7: Model Architecture (Dueling DQN)

Layer	Size / Details
Input Layer	Shape: (1, `state_dim`)
Flatten	Converts input to 1D
Dense Layer 1	24 units, ReLU activation
Dense Layer 2	24 units, ReLU activation
Value Stream	Dense (1 unit), Linear activation
Advantage Stream	Dense (num_actions), Linear activation
Merge (Q-Value Output)	$Q(s,a) = V(s) + (A(s,a) - \frac{1}{\lvert A\rvert} \sum A(s,a'))$

To ensure a smooth transition in difficulty, we employed a curriculum learning strategy where the enemy bot's behavior gradually shifts from random to more skillful, heuristic-based movement. The full training setup and curriculum strategy are described in Table 8.

Table 8: Training Hyperparameters (Dueling DQN with Curriculum Training)

Parameter	Value / Details
Environment Size	400×400
Tank Health	3
Action Space Size	8 actions
Episodes (Steps)	5,000,000 steps
Test Episodes	5
Replay Memory	Limit = 5,000,000; window_length = 1
Warmup Steps	100
Target Model Update	0.01 (soft update)
Double DQN	Enabled
Exploration Policy	Linear Annealed Epsilon-Greedy
	ϵ: decays from 1.0 to 0.05 over 10,000 steps
	Test $\epsilon = 0.05$
Policy	Epsilon-Greedy
Memory Type	SequentialMemory
Optimizer	Adam
Learning Rate	0.005
Model Architecture	Dueling DQN with [24, 24] hidden layers
Loss Function	Mean Squared Error (MSE)
Callback	Custom signal-safe logger and best weight saver
Curriculum Learning	Gradual increase in TrainingBot skill over time

Finally, we created two reward functions for the different models. Table 9 discretely rewards results that lowers enemy's health and punishes results that lowers the player's health, while rewarding defeating the enemy and punishing losing to the enemy. Table 10 is more granular in its feedback by rewarding the player for aiming at the target or shooting at the target proportionally to the closeness of the aiming and shooting. Since this aiming and shooting mechanism is continuous, the reward/punishment for winning/losing the game is consequently scaled up.

Table 9: Reward Function Components for DQN Model (Experiment 2)

Condition	Reward / Penalty
Player Hits Enemy	$+50$
Player Gets Hit by Enemy	-50
Winning Bonus (Scaled by Health)	$+10\times$ Player Health
Losing Penalty (Scaled by Time & Enemy Health)	$-(10\times$ Total Enemy Health$+$ $10 \times (1 -$ train_time factor$))$

Table 10: Reward Function Components for Dueling DQN Model (Experiment 2)

Condition	**Reward / Penalty**
Player Hits Enemy	$+5000$
Player Gets Hit by Enemy	-5000
Player's Barrel is x Degree Away From Enemy*	$-tanh(5 \times (0.5 - \frac{x}{180}))$
Player's Bullet is Shot x Degree Away From Enemy*	$-tanh(5 \times (0.5 - \frac{x}{180}))$
Winning Reward (Scaled by Time)	100((Player Health $\times$10) $-1.0\times$ Steps)
Losing Penalty (Scaled by Time)	-100((Enemy Health $\times$ 10) $-$ 1.0$\times$ Steps)

This structured design across both experiments allows us to incrementally build, evaluate, and analyze the effectiveness of various learning strategies and model architectures in adversarial tank battle scenarios.

4 Results

4.1 Experiment 1 Results

The results from Experiment 1 indicate that the agent struggled to effectively learn the task of aiming and shooting at a stationary target. As shown in Figure 2, the training reward curve remains largely flat throughout the episodes. While there is a slight upward trend in total reward between episodes 0 and 100, this improvement is marginal and does not suggest meaningful learning progress. Overall, the lack of a consistent increase in total reward implies that the agent was unable to reliably develop a successful aiming policy in this simplified environment.

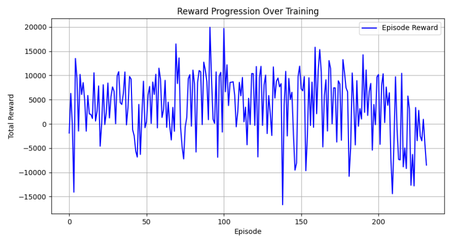

Figure 2: Training reward per episode for Experiment 1. The curve remains mostly flat, with a slight upward trend early on.

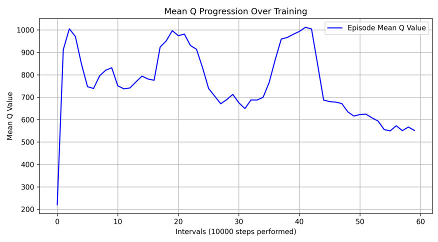

Figure 3: Mean Q-value estimates over training. Early gains are followed by volatility and gradual decline.

Figure 3 presents the mean Q-value estimates throughout training. The plot exhibits an initial sharp increase during the early training intervals (0–3), but is followed by a series of peaks and a gradual downward trend, suggesting potential instability in value estimation as training progresses.

Training footage of the DQN agent (12), including examples of turret rotation, periodic shooting, and aiming behaviors, is available at:

`https://www.youtube.com/watch?v=L1F0oICzcog`

4.2 Experiment 2 Results

4.2.1 Dueling DQN Model

The results from Experiment 2 indicate that while the agent demonstrated some improvement in handling both aiming and movement, its learning progression was highly unstable, as seen in the fluctuating reward curve (Figure 4). Unlike Experiment 1, where rewards remained relatively flat, Experiment 2 exhibits high variability in episode rewards. This suggests that while the agent was able to achieve periods of improved performance, it struggled with consistency. The fluctuations likely stem from the increased complexity of balancing offensive (aiming/shooting) and defensive (dodging) actions, which may have led to frequent changes in strategy as the agent explored different behaviors.

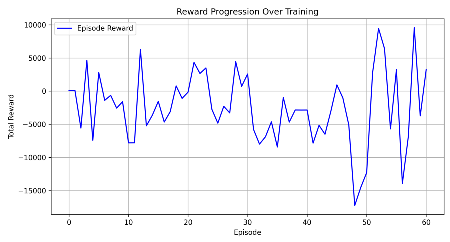

Figure 4: Training reward per episode of the Dueling DQN Model for Experiment 2. The curve shows more fluctuation but an overall upward trend.

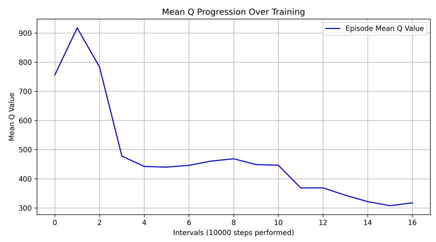

Figure 5: Mean Q-value estimates of the Dueling DQN Model during training. Initial gains are followed by noise and slight decline.

Figure 5 presents the mean Q-values across training intervals. The plot initially shows a sharp increase in Q-values, indicating that the agent rapidly discovered high-value actions early in training. However, this is followed by a notable decline and long-term instability, suggesting that the agent struggled with Q-value estimation. This may indicate overestimation issues or instability in policy convergence due to the dynamic nature of the environment.

4.2.2 Standard DQN Model

Since the Dueling DQN agent was unable to be run on our available machine for extended periods of time, we down-scaled the model by training a trivial DQN agent with the same parameter for the Input, Flatten, Dense Layer 1, Dense Layer 2, but with a third dense output layer that outputs to the action space with a linear activation. This model showed significant improvement in strategy after 60 iterations (60000 steps) of training, exhibiting clear learned behavior of dodging oncoming bullets.

A demonstration of our trained Standard DQN agents (13) in the tank battle environment is available at:

`https://www.youtube.com/watch?v=2z1KS27iiSk`

4.3 Performance Results

To assess the effectiveness of our reinforcement learning agent, we conducted a series of matchups against different opponents, including a random policy, a heuristic-based opponent, and human players. Figure 6 presents the number of wins recorded for both the left and right opponents in various matchups. Since early exploration of our heuristic algorithm proved itself to be too powerful, the heuristic model used in evaluation takes one heuristic step for every 100 random steps.

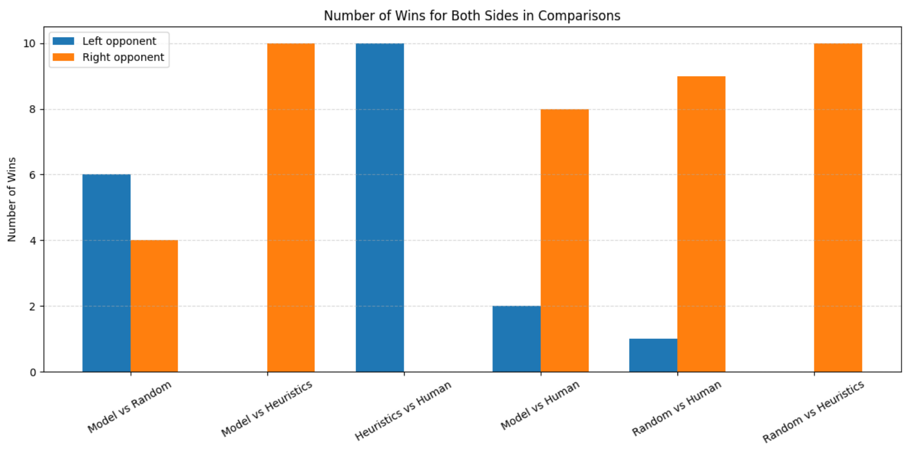

Figure 6: Number of wins recorded across different matchups. The left opponent (blue) represents the first agent, while the right opponent (orange) represents the second agent in each matchup.

- **Model vs. Random Opponent:** The reinforcement learning agent outperformed the random opponent, securing 6 wins compared to the random bot's 4 wins. This suggests that the trained model learned some degree of strategic behavior.
- **Model vs. Heuristic Opponent:** The model struggled against the heuristic bot, losing all 10 matchups. This highlights the difficulty in overcoming a rule-based adversary with predefined movement and shooting strategies.
- **Heuristic vs. Human:** The heuristic agent demonstrated strong performance against human players, winning 8 out of 10 games.
- **Model vs. Human:** The reinforcement learning model performed poorly against human opponents, winning only 2 out of 10 games.
- **Random vs. Human:** The random agent performed the worst, securing only 1 win while humans won 9 times.
- **Random vs. Heuristic:** As expected, the heuristic agent consistently outperformed the random agent, winning 10 out of 10 games.

The results indicate that the reinforcement learning agent was able to outperform a random strategy, demonstrating that it successfully learned some useful policies. However, the heuristic agent remained the strongest competitor, suggesting that rule-based strategies still outperform learned policies in this environment. Human players consistently outperformed both the model and the random agent but had more difficulty against heuristic opponents, which suggests that the heuristic bot's strategies align more closely with human intuition. Additionally, the reinforcement learning agent struggled against human opponents, likely due to its inability to generalize effectively to unpredictable strategies.

These findings highlight the potential of reinforcement learning to develop competitive strategies in structured environments but also reveal key limitations. Future improvements such as curriculum learning, better reward shaping, or advanced architectures (e.g., Proximal Policy Optimization (PPO) and CNN-based encoders) may be necessary to enhance the agent's ability to generalize against more adaptive and complex opponents.

5 Discussion

5.1 General Discussion

Our results were expected matches to our available architecture.

A key observation across both experiments was the instability in learning. While Q-value estimates initially increased as expected, later training stages exhibited fluctuations, particularly in Experiment 2. This suggests that either the exploration policy or Q-value estimation suffered from inconsistencies, potentially caused by the agent's difficulty in predicting long-term rewards in a dynamic adversarial setting.

5.2 Experiment 1: Stationary Aim and Shoot

The mean Q-value progression graph shows a high initial Q-value of around 1000, followed by a decline that stabilizes between 500 and 600. This pattern is expected in reinforcement learning, where the model initially overestimates rewards and then gradually corrects itself through experience. However, the sharp fluctuations throughout training indicate that the DQN is making large updates, likely due to high variance in rewards. This is consistent with our qualitative observations in the training stage, where the model is rewarded heavily for aiming and shooting the target, but not sufficiently punished for time wasted, leading to peaks of reward disproportional to the performance.

One concerning trend is the drop in Q-values near the later stages of training. This could suggest that the agent is either becoming too conservative (avoiding exploration). Epsilon decay has reduced exploration, causing the agent to repeatedly use suboptimal but "safe" strategies, such as waving its barrel in the right quadrant instead of aiming at the opponent and shooting. We are interested to see if increased episodes would exhibit the current oscillating behavior, or perhaps stabilize. The former would suggest that the model architecture and parameters simply is not sufficient to learn the task, while the latter would suggest the poor learning is caused by a lack of training experience.

The reward progression graph presents high variability, with total episode rewards oscillating between high positive values (15,000 to 20,000) and extreme negative values (-15,000). In early training, these fluctuations are expected due to the agent's exploration phase, where it experiments with different actions to understand the environment. However, as training progresses, the reward values remain unstable, suggesting that the agent is struggling to establish a consistently effective strategy within the available training period.

Around episode 100, there is a visible increase in reward peaks, which suggests the agent found a successful strategy. However, later episodes show a declining trend, where rewards start trending downwards again. Observing the training footage, we found that the model continues to learn and even exhibit behaviors that suggest learning and memorization of the opponent's location up till the 50000th step; the model would gradually rotate its barrel towards the opponent while shooting bullets periodically, and once the bullet hits the opponent it then waits for the shooting cool down before shooting again. (Supplemental Media (12); 1:05:45) However, there is a notable decline in performance in the last 10000 steps that may show the model becoming too conservative, as mentioned above, and the aforementioned behavior disappears along with decreasing reward.

5.3 Experiment 2: Dynamic Aim, Shoot, and Dodge

The introduction of movement in Experiment 2 significantly increased task complexity. This is evident by the sharp reduction in episodes our machine was capable of training the dueling DQN model.

5.3.1 Dueling DQN Model

The mean Q-value progression graph showed an initial increase followed by a sharp decrease and a slow and gradual decrease until the end of training. The steep decline in early training steps could be due to the agent realizing that actions deemed high value early on did not reward itself as expected, which lead to an adjustment in policy. The continuous decrease towards the end could indicate that the model is unable to find consistently optimal strategy, and unlike the previous model with less parameters that had periods , does not have sufficient representation capable of forming such strategy.

The reward progression graph presents high variability as did in Experiment 1. The oscillation increased alongside episode count, suggesting that the model is still heavily exploring strategies in the 45th-60th episodes. Observational data supports this as the model showed no improvement in its strategy throughout training; the movement is not jittery like a random agent, but there was no clear demonstration of search and aim strategy combined with dodging strategy. It has occasional episodes of clarity where it shoots in the direction of the opponent while continuously rotating its barrel clockwise, but overall showed limited improvement.

5.3.2 Standard DQN Model

Contrasting the poor performance of the dueling DQN model, the base DQN model surprised us with the ability to dodge incoming bullets through moving or rotating its body.(Supplemental Media (13)) There are two possible explanations to this surprising performance. The first and the most trivial is that since the model is able to train for longer, it was given time to develop its first strategy. The second explanation is that the differing reward function emphasizes defensive and offensive actions equally, thereby enabling the model to choose one specifications over the other unlike the offense focused reward function in the dueling DQN model. However, this model exhibits a near-random behavior in its offense, therefore rarely wins games. Nevertheless, a model of this size exhibiting dodging strategy within 60000 steps of learning is impressive.

5.4 Miscellaneous Experiments (PPO, Imitation Learning)

During the course of our experiments, we tested additional reinforcement learning techniques beyond Deep Q-Learning (DQL) to determine whether alternative methods could improve performance in our tank battle environment. While these methods showed some promise, they did not yield consistently strong enough results to be included in the main evaluation.

We experimented with Proximal Policy Optimization (PPO) as an alternative to DQL, given its effectiveness in continuous control tasks and policy-based reinforcement learning. The motivation behind testing PPO was its ability to optimize policies directly and handle more dynamic action spaces efficiently. However, in our implementation, PPO struggled with learning an effective strategy, particularly in Experiment 2. The policy updates often resulted in erratic behavior, and the training process was significantly less stable than with DQL. This could be due to the sparse and delayed rewards in our environment, which PPO may not have been well-suited to handle without additional reward shaping or statstate-encodinghniques.

We also attempted Imitation Learning, leveraging scripted heuristic agents to generate expert demonstrations. The goal was to pre-train the reinforcement learning agent using behavior cloning or by incorporating expert trajectories into experience replay. However, despite initializing the model with expert actions, the agent failed to generalize effectively beyond the limited scenarios in the demonstrations. This suggests that the imitation learning process lacked sufficient diversity in training examples, leading to overfitting to specific situations rather than developing robust, adaptive behaviors.

Although PPO and Imitation Learning did not yield sufficiently strong results in this study, they remain promising avenues for future research. Further tuning of hyperparameters, architectural modifications, and more sophisticated reward functions may improve the effectiveness of these techniques in adversarial combat scenarios.

5.5 Limitations

One of the primary limitations of our study was the lack of consistent access to GPU-accelerated compute resources. Much of the training was conducted on local machines, which significantly constrained the number of training episodes and the complexity of the experiments we could reasonably perform. As a result, we were unable to run extended hyperparameter sweeps, train larger models, or fully explore alternative architectures such as convolutional or transformer-based networks.

This computational bottleneck likely contributed to some of the training instability observed in Experiment 2, as we were unable to train long enough to achieve convergence or to reliably distinguish between noise and meaningful learning. Additionally, limited hardware resources meant that we could not run and replicate each experiment multiple times to assess variance in performance.

Although we implemented curriculum learning and tuned our reward structure to mitigate some of these constraints, future iterations of this work would benefit from access to cloud-based GPU infrastructure or university cluster resources to support larger-scale experimentation and more thorough evaluation.

6 Future Work

We were not able to train the model for extended periods under multiple different conditions due to the time restraint; future work focused on aim, shoot, and dodge games could consider the following factors when designing their experiment.

Firstly, Experiment is a complex 2D task with a minimum of 3 possible movements (as seen in Experiment 1), with moving targets between games. Let's compare this to the DQN model that plays the game *Breakout* in Mnih(2013): its reward and q value peaked at around episode 50, both then followed a sharp decline before increasing again later and eventually reaches good performance at 100 episodes. Our DQN agent demonstrates an initial increase in efficiency throughout the first 50 episodes, but the test result yields marginally better movement that does not meet the goal of the game. While using very different architecture, we can observe that at least 100 episodes is needed, as seen in Mnih(2013), in order for the model to show human level improvement at the level tasks demanded by Experiment 1.

Secondly, Experiment 2 is a complex 2D task with 8 possible movements, sectioned into control of 2 components of an entity. This task has proven to be extremely difficult with the current architecture: without CNN and with a minimally sized DQN. In the future, it is worth expanding the architecture to first pass the frame/state through a CNN to extract meaningful representations before feeding it into the DQN to determine an action.

Lastly, alternatively determined reward may be helpful to the model's learning. Since the game's scoring system does not match its complexity that is required for a model's learning (score is determined without considering the closeness of shots or number of dodges), there might be a better reward system than the one we have generated heuristically.

7 Authors' Contributions

Sean - One of the collaborators on base DQL model, and performed debugging and tuning through group programming. Worked on the corresponding portion of the project report. Worked on writing code to collect human baseline for both experiments all while taking the highest amount of final exams in the team. Peer programmed with Wilson, Avi, Hikaru. I affirm all team mates contributed to coding and contributed equally.

Avi - One of the collaborators on writing the reward method, one of the collaborators on the fine-tuned DQN models. Worked on the corresponding portion of the project report. Highest frequency of claims of "crashing out" despite maintaining morale. Volunteered to be human data. Coded up auto-aim for agent, which sadly did not make report. Peer programmed with Wilson, Julia, Hikaru. I affirm all team mates contributed to coding and contributed equally.

Wilson - Wrote the base Tank game, wrote the base DQN model and the imitation learning model. Wrote a heuristic bot that is impossible for both humans and machines to beat. Coded up a model with PPO and imitation learning, which also sadly did not make report. Peer programmed with Julia, Avi, Hikaru. I affirm all team mates contributed to coding and contributed equally.

Julia - Wrote and trained the dueling model, one of the collaborators on writing the reward method, wrote visualization and testing code. For all portions mentioned, collaborated with teammates to write the report. Brought coffee and a broken datahub account. Helped debug bits and pieces of all sections of code. Peer programmed with Wilson, Avi, Hikaru. I affirm all team mates contributed to coding and contributed equally.

8 Acknowledgments

We kindly thank the CSE 151B/251B instruction team in supporting our learning throughout the quarter, without which we would not have done this project. We also thank our mentor Zheng Ding for his guidance and brilliance.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[2] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search.

[3] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning.

[4] Schaul, T., Quan, J., Antonoglou, I., Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

[5] Van Hasselt, H., Guez, A., Silver, D. (2016). Deep reinforcement learning with double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.

[6] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning*.

[7] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[8] Zhang, J., Agarwal, A. (2020). Augmented Q Imitation Learning. *arXiv preprint arXiv:2004.00993*.

[9] Reddy, S., Dragan, A. D., Levine, S. (2019). SQIL: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*.

[10] He, L., Zhang, X., Sun, M. (2022). Heuristic-Guided Reinforcement Learning for Constrained Decision-Making. *Proceedings of the AAAI Conference on Artificial Intelligence*.

[11] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., ... Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

[12] Wong, J. (2025). Training Footage DQN Dueling Model [Video]. YouTube. `https://www.youtube.com/watch?v=L1FOoICzcog`

[13] Mehta, A. (2025). Footage Experiment 2 [Video]. YouTube. `https://www.youtube.com/watch?v=2z1KS27iiSk`